Technical Notes

Predictive Analytics Solution for ATM Cash Transactions

KBSL Information Technologies Limited

Prepared for
National Development Bank

Revision History

Version	Date	Summary of Changes	Author	Revision Marks (Yes/No)
0.1		Initial revision.		
0.2		Second Revision		

Table of Contents

Predictive Analytics Solution for ATM Cash Transaction Release Notes

1. Introduction

The document presents the major new features which have been accommodated in this release of the Predictive Analytics Solution for ATM cash transaction– Version 0.1. It also discusses identified glitches and remedial actions.

2. About This Release

This release, Version .1 demonstrates the process of setting up live environment for production. The below mentioned main steps should be covered under this release.

- Setting up server which front end application runs
- Setting up server which back end scripts runs

3. New Releases

3.1 Setting up server which front end application run:

3.1.1. Setup non root user:

Setting up dev tools python package and pip package

1. sudo yum install epel-release
2. sudo yum install python-pip python-devel gcc nginx
3. sudo yum install python3-devel openldap-devel
4. sudo pip install virtualenv

3.1.2 Setting up working directory same as Git clone

1. mkdir ~/ NDB-FrontEnd
2. cd ~/ NDB-FrontEnd

3.1.3 Cloning the Git repository

1. git clone https://github.com/Kbsl-data-labs/NDB-FrontEnd.git
2. git add remote

3.1.4 Creating virtual environment and install relevant packages to the virtual environment

1. Virtualenv ATM360Front
2. source ATM360Front /bin/activate
3. pip install django gunicorn django-tables2 tablib django-crispy-forms numpy pandas plotly django-auth-ldap python-ldap

3.1.5 Django configurations

1. Cd \NDB-FrontEnd\django_adminlte3\settings.py
Keep as ALLOWED_HOST=[*]

2. Change LDAP Authentication accordingly.
* Please refer Annexure I.

3. Uncomment STATIC_ROOT line in settings.py file
STATIC_ROOT = os.path.join(BASE_DIR, "static")

4. Set up backend URLs as follows
cd NDB-FrontEnd\adminlte3\helpers\api_helpers.py
change login URL and ATM URL for their respective back end URLs.

5.   Completing initial project setup
Change directory where the manage.py script is located. Type below commands
Python manage.py makemigrations
Python manage.py migrate
Python manage.py createsuperuser

(Give super user name and password)
Python manage.py collectstatic

3.1.6 Setting up Gunicorn server

1. Testing ab   ability to serve the project
   cd NDB-FrontEnd\ NDB-FrontEnd
   gunicorn --bind 0.0.0.0:8000 django_adminlte3.wsgi:application

2. Deactivate the virtual environment by typing “deactivate” command

3.1.7 Creating system socket file and service file for gunicorn

1. sudo vi /etc/systemd/system/gunicorn.service
2. Include Systemd gunicorn service file (Figure1.1) content to the service file.
3. Set the user and user group accordingly. (Replace your user and group name in user and group line)
4. Enale the gunicorn service:
   sudo systemctl start gunicorn
   sudo systemctl enable gunicorn

3.1.8 Setting up nginx server by modifying nginx.conf file

1. Change nginx.conf file as Figure 1.2
sudo vi /etc/nginx/nginx.conf

2.Change server IP accordingly after the server name line.

```
[Unit]
Description=gunicorn daemon
After=network.target

[Service]
User=user
Group=nginx
WorkingDirectory=/home/user/NDB-FrontEnd
ExecStart=/home/user/NDB-FrontEnd/ATM360Front/bin/gunicorn --workers 3 --bind
unix:/home/user/NDB-FrontEnd/NDB-FrontEnd.sock NDB-FrontEnd.wsgi:application

[Install]
WantedBy=multi-user.target
```

Figure 1.1 – Systemd gunicorn service file

```
server {
    listen 80;
    server_name server_domain_or_IP;

    location = /favicon.ico { access_log off; log_not_found off; }
    location /static/ {
        root /home/user/NDB-FrontEnd;
    }

    location / {
        proxy_set_header Host $http_host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_pass http://unix:/home/user/NDB-FrontEnd/NDB-FrontEnd.sock;
    }
}
```

Figure 1.2: NGINX.conf file

3. Change user mode group permission to nginx group
   sudo usermod -a -G user nginx

4. Give our user group to execute permission on our /home directory. Since project front end content saved under home directory.
   chmod 710 /home/user

5. Test your nginx configuration file for syntax errors
   Sudo nginx –t

6. Start and enable the nginx process
   sudo systemctl start nginx
   sudo systemctl enable nginx

7. Try to log on to the application in chrome browser by typing http://server_IP

3.2 Setting up server which back end scripts run:

3.2.1 Setting up Mongo DB in back end server

1. Configure package management system

Vi /etc/yum.repos.d/mongodb-org-4.4.repo

Add content in Figure 1.3 to this file

```
[mongodb-org-4.4]
name=MongoDB Repository
baseurl=https://repo.mongodb.org/yum/redhat/$releasever/mongodb-org/4.4/x86_64/
gpgcheck=1
enabled=1
gpgkey=https://www.mongodb.org/static/pgp/server-4.4.asc
```

Figure 1.3: Mongo DB repo file

2. Install Mongo DB packages

sudo yum install -y mongodb-org

3. Alternatively, to install a specific release of MongoDB, specify each component package individually and append the version number to the package name by typing following command

sudo yum install -y mongodb-org-4.4.0 mongodb-org-server-4.4.0 mongodb-org-shell-4.4.0 mongodb-org-mongos-4.4.0 mongodb-org-tools-4.4.0

4. To prevent unintended upgrades, pin the package. To pin a package, add the following exclude directive to your /etc/yum.conf file:

exclude=mongodb-org, mongodb-org-server, mongodb-org-shell,mongodb-org-mongos,mongodb-org-tools

6. By default, MongoDB runs using the mongod user account and uses the following default directories:

   /var/lib/mongo (the data directory)

   /var/log/mongodb (the log directory)

7. By default, MongoDB runs using the mongod user account. Once created, set the owner and group of these directories to mongod:
   sudo chown -R mongod:mongod <directory>

3.2.2 Setting up Mongo DB dump in back end server

1. Start and enable mongod service

Sudo systemctl start mongod

Sudo systemctl enable mongod

Sudo systemctl status mongod

2. If mongod service is running, go to mongo shell installation directory

Cd <Mongo shell installation directory>

Type command mongo

3. Create a database in mongo db

Use <your database name>

4. Restore your dump file to database

mongorestore --db=<Database name> --collection=<Collection name> <Location where dump bson file saved>.bson

3.2.3 Setting up application back end directory

1. Creating application back end directory.

```
Cd /home/ATM-360-dev
```

2. Cloning the git repository

```
git clone https://github.com/Kbsl-data-labs/ATM360.git
git add remote
```

3. Create virtual environment using pipenv command

```
 Cd /home/ATM-360-dev/envs
 Pipenv shell
```

4. Install relevant packages to the virtual environment

```
Pipenv install falcon gunicorn falcon_auth pymongo
```

5. Start the back end service by typing following command:

```
Cd /home/ATM-360-dev/ ATM-360-dev
gunicorn app:api –b server_ip:port
```

3.3 Running Back Ground Service Files

3.3.1 – To Data Base Service

To start, stop or restart the data base service, run below commands in **sudo user mode/root user mode.**

```
systemctl start ATM360_DB
systemctl stop ATM360_DB
systemctl restart ATM360_DB
```

3.3.2 – If ATMOsphere Application Fails

First check whether the process ID of “gunicorn” process currently running in the server/ whether “gunicorn” process is running or not by typing below command

```
Ps –ax | grep gunicorn or ps –ax
```

Then kill the running process by:

```
Pkill gunicorn or pkill (process ID)
```

Then restart the application service file by running below command

```
Systemctl restart ATM360_APP
```

If you need to start or stop the application use below commands:

```
Systemctl start ATM360_APP
Systemctl stop ATM360_APP
```

4. New Features

The following new feature will appear in this release:

1. Frond End application and backend scripts are running in live environment
2. Mongo DB installed in backend server
3. Application can be accessed via http://Server_IP

5. Identified Glitches and Limitations

5.1 General Note

This release will set up the front end and back end servers in live environment including mongo db.

.

Annexure I- LDAP Configuration

1. Change lines in settings.py file accordingly

   Connect.Simplebind_S("AP_ATMOsphere@ndlk.int","#$ATOS2020!@")
2. Uncomment lines as follows:

```
# LDAP authentications
# Use the custom AbstractUser table for LDAP authentication
AUTH_USER_MODEL = 'ldapAuth.User'
LOGIN_URL = '/ldapAuth/login/'
# LDAP configuration
import ldap
from django_auth_ldap.config import LDAPSearch,
GroupOfNamesType,LDAPSearchUnion,LDAPGroupQuery



# # Baseline configuration.
AUTH_LDAP_SERVER_URI = "ldap://10.2.2.10:389"

AUTH_LDAP_BIND_DN = "AP_ATMOsphere@ndblk.int"
AUTH_LDAP_BIND_PASSWORD = "#$ATOS2020!@"
AUTH_LDAP_USER_SEARCH = LDAPSearchUnion(
    LDAPSearch("ou=NDB Users,dc=ndblk,dc=int", ldap.SCOPE_SUBTREE,
"(sAMAccountName=%(user)s)"),
    #LDAPSearch("ou=Region-01,dc=kbsl,dc=lk",ldap.SCOPE_SUBTREE,
"(sAMAccountName=%(user)s)"),
    #LDAPSearch("ou=CPU,dc=kbsl,dc=lk",ldap.SCOPE_SUBTREE,
"(sAMAccountName=%(user)s)"),
)
# # Or:
# # AUTH_LDAP_USER_DN_TEMPLATE = 'uid=%(user)s,ou=users,dc=example,dc=com'
# # Set up the basic group parameters.
AUTH_LDAP_GROUP_SEARCH = LDAPSearchUnion(
    LDAPSearch("ou=Application Groups,ou=Applications,dc=ndblk,dc=int",ldap.SCOPE_SUBTREE,"(objectClass=groupOfNames)"),

#LDAPSearch("cn=ATMTEST,ou=DataLab,dc=kbsl,dc=lk",ldap.SCOPE_SUBTREE,"(objectClass=groupOfNames)"),
)


AUTH_LDAP_GROUP_TYPE = GroupOfNamesType(name_attr="cn")


# # # Simple group restrictions
#AUTH_LDAP_REQUIRE_GROUP ="cn=Datalab,ou=DataLab,dc=kbsl,dc=lk"
# # AUTH_LDAP_DENY_GROUP = "cn=disabled,ou=django,ou=groups,dc=example,dc=com"


# # Populate the Django user from the LDAP directory.
AUTH_LDAP_USER_ATTR_MAP = {
     "first_name": "givenName",
     "last_name": "sn",
#    #"email": "mail",
 }


AUTH_LDAP_USER_FLAGS_BY_GROUP = {
     "is_active": "cn=AP-ATM360-User-DL,ou=Application Groups,ou=Applications,dc=ndblk,dc=int",
     "is_region": "cn=AP-ATM360Regional-CPUCashUsers-DL,ou=Application Groups,ou=Applications,dc=ndblk,dc=int",
     "is_staff": "cn=AP-ATM360Headofficestaff-DL,ou=Application Groups,ou=Applications,dc=ndblk,dc=int",
# #     "is_superuser": "cn=superuser,ou=django,ou=groups,dc=example,dc=com",
 }


# # This is the default, but I like to be explicit.
AUTH_LDAP_ALWAYS_UPDATE_USER = True
```

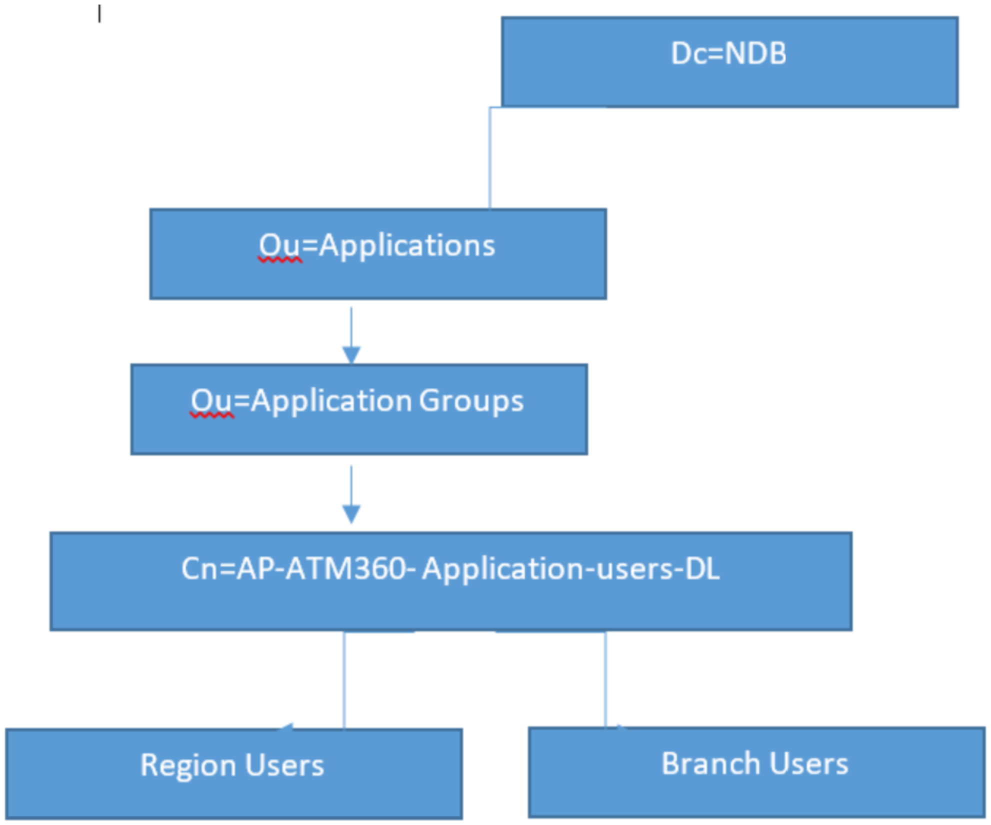
```
# # Use LDAP group membership to calculate group permissions.
AUTH_LDAP_FIND_GROUP_PERMS = True

# # Cache distinguished names and group memberships for an hour to minimize
# # LDAP traffic.
# AUTH_LDAP_CACHE_TIMEOUT = 3600

AUTHENTICATION_BACKENDS = [
    "django_auth_ldap.backend.LDAPBackend",
    "django.contrib.auth.backends.ModelBackend",
]

STATICFILES_DIRS = [
   os.path.join(BASE_DIR, "static"),
]
#STATIC_ROOT = os.path.join(BASE_DIR, "static")
STATIC_URL = '/static/'

CRISPY_TEMPLATE_PACK = 'bootstrap4'
```



LDAP Structure in AD